▨ 연구논문

Systems Modular Approach For Design and Analysis of Object Oriented Simulation Software

객체지향 시뮬레이션 소프트웨어의 설계 및 분석을 위한 시스템 모듈식 접근방법에 관한 연구

Yoo, Wang Jin*
유 왕진
Lim, Ik Sung**
임 익성
Kim, Tae Sung**
김 태성

요 지

최근까지 개발된 시뮬레이션 소프트웨어를 검토 정리한 후, 시뮬레이션 소프트웨어와 그 모델과의 연관성을 파악하기 위하여 총체적인 시뮬레이션 모델을 관찰하였으며, 시뮬레이션 소프트웨어의 유연성에 대한 근원을 시스템 모듈식 접근 방법을 통하여 추적하였다. 실존 시스템, 모델, 그리고 소프트웨어 시스템과의 관계를 조사한 결과 시뮬레이션 소프트웨어의 주요한 성능척도는 flexibility와 accuracy라는 것이 밝혀졌다. 객체지향 시뮬레이션 소프트웨어의 metrics와 formalism의 모델을 개발하였으며, 이것은 유연한 객체지향적 시뮬레이션 소프트웨어 구조를 디자인하는데 근본 방침을 제시한다. 끝으로 앞에서 개발한 모델을 기초로, 유연한 객체지향적 시뮬레이션 소프트웨어 시스템을 분석하였다.

I. Introduction

In order to meet the user's demands, varieties of simulation software have been continuously developed from the late 1950's. Currently, the number of simulation languages (including academic and commercially available) are known to over 140 [16]. Basically, there are two types of simulation softwares (or languages in a broad sense): (i) Discrete event simulation model and (ii) Continuous simulation model [12, 14, 19].

1.1 Overview of simulation languages

Discrete event models are described by logic, rules and procedures. These languages include GPSS, ARENA/SIMAN IV, SLAM II/TESS, Simula, GEMS-II, CVASD, INSIGHT, and SIMSCRIPT II [15]. Continuous simulation model concerns the modeling of systems over time by representing state variables changes continuously. In continuous simulation, models are usually described by difference/differential equations. Some of these languages include Extend, Model-it, SIMULAB, MATLAB, STELLA, DYNAMO, CSMP, ACSL, MDOF, PIPEPHASE, TPS, GVS, TUTSIM, SIMNON, DARE, Modeller, MATRIX, ESL, GEMS, ECAP, and PCAP [10,12,17,19].

* Kon-Kuk University, Dept. of Industrial Engineering
** Namseoul University, Dept. of Industrial Engineering

However, since real systems under consideration are neither completely discrete nor completely continuous, models are needed that combine discrete event and continuous models. Thus, the combined discrete event and continuous languages were developed in order to provide more power (note: power indicates the ability of simulation software to build an arbitrary model). These include WITNESS, Extend, DEVS-Scheme, ARENA/SIMAN IV, COSY, ModSim II, SYSMOD, PCModel, HOCUS, SLAM II, DISCO, GASP IV, COSMOS, CLASS, PROSE, CDCSIS, NEDIS, and GSL [1,2,5,13,15,21,16].

In order to reduce the user's effort for building a model, simulators, advertised as "no-programming required" or "ease-to-use" were developed. These simulators (sometimes called Special Purpose Simulation Software or application oriented simulation languages) include BETHSIM, SmartSim, XCELL+, WITNESS, QUEST, Micro Saint, COMNET, Net Work II, MAP/1, MAST, Speed, See-Why, Modelmaster, BONeS, FACTOR, INSIGHT, JR-net, SimFactory, SIMPLE++, AutoMod, ProModel, and TAYLORII [1,15, 22, 24].

Since then, there have been trends towards object orientation due to the number of benefits of object oriented languages. The advantages of using object-oriented languages such as Eiffel, Smalltalk, Simula, C++, and Objective-C can be found from Najmi [20, 25]. The object-oriented simulation languages include SmartSim, Sim++, SIMPLE++, CSIM, MB, SimKit, SimPack, Ross, INSIGHT, ConSim, Extend, DISCO, ModSim II, and DEVS-Scheme [5, 6, 7, 16, 18, 23, 25, 27].

Simulation software has been successful in satisfying the users to some degree. However, there is always a trade-off between power and ease-of-use in simulation software. For example, the commercially available simulators allow the user to build a domain specific model with ease-of-use but less capability to build different domain models. On the other hand, the general purpose simulation software allows the users to build a wide range of domain models, which is called power, but it requires more effort than the simulators.

The main reason for the existence of trade-off is due to the fact that simulators are designed for particular domain problems. However, in fact, the user's environment is constantly changing. For example, if the modeler wants to model a different domain problems, then the modeler has to devote significant effort to build a different domain model. Therefore, a major issue becomes maintaining the ease-of-use while increasing the power. The clue to the solution of this problem can be found in the concept of 'flexibility', which has been used in the manufacturing systems area. Simulation software flexibility can be thought as "the ability of simulation software systems to respond to changing requirements."

2. Analysis of object oriented simulation software

Real systems under study cover a wide range of sizes, structures, and behaviors. For different types of real systems, the modeler needs to build a models to satisfy the different system requirements. Therefore, according to the concept of flexibility, flexible simulation software systems should allow the modeler to model the different system requirements with minimal effort.

2.1 Definitions

Terminology and definitions used in this research may be somewhat different from those used by other researchers so they need to be clearly explained.

* Entity (ENT) is an object engaged in activities and characterized by its data values called attributes. It also has relationship(s) and behavior(s).

* Structure(STR) is the arrangement of relationships between modules. Therefore, it is composed of modules and relationships.

* Behavior (BHR) describes the possible actions of an entity through its activities. In a sense, behavior can be considered to a modular, module so that several actions can be grouped into higher level activities.

* Attribute (ATT) describes the state of an entity whose characteristics are expressed via data values.

* Relationship (REL) constitutes "awareness" by one entity of another (the related entity).

* Interface (IRF) is a channel through which other entities can communicate with an entity. It contains constraints such as data types and acceptable data values that can be communicated.

2.2 View of object oriented simulation model

Based on the above definitions and terminology, a simulation model can be viewed as shown in Figure 1. A simulation model is composed of structure and behavior. Structure is composed of relationships and entities, and entities have relationships, behavior and data. Behavior of entities can be described by either logical constructs and/or differential equations using operators/operands. The former case is usually called discrete event model and the latter is called a continuous model. Two different types of entities can be present in a model. A permanent entity is always present in a model, while a transient entity is dynamically created and then destroyed during a simulation study.

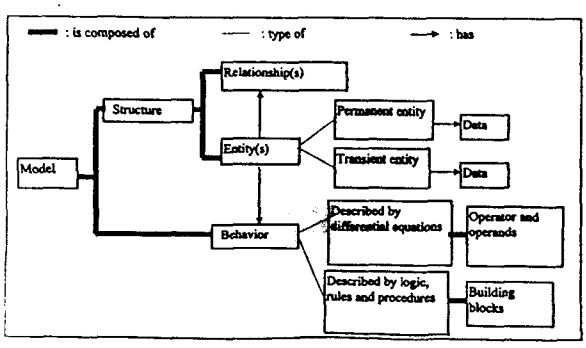

Figure 1. A generic view of simulation model.

3. Formalism and metrics of object oriented simulation software

A real system of class c is expressed as Sc which is composed of modules.

S_c = { M'i } , where, M'i is a set of modules which are the real system's component objects. Furthermore, Sc has certain modeling requirements (RQ) which include the type of structure and behaviors in such systems. The modeling requirements of a system of class c is expressed as $RQ(S_c)$.

A modular module Mi , object oriented, has behaviors BHR, relationships REL, attributes ATT, and interfaces IRF.

Mi = < BHR, REL, ATT, IRF >

A simulation model of a class c system is expressed as SMc, and is composed of modules Mi.

SM_c = { $(M_1, M_2, ..., M_n)$| i∈ integer }

A simulation software system for modeling class c systems, SS_c, is configured to produce SM_c in order to meet $RQ(S_c)$ with as much ease-of-use as possible as well as accuracy relative to S_c. SS_c can be thought as a collection of functionality for transforming and modeling the real systems. Thus, SSc has a set of transformation(TMF_i) and modeling functions for abstracting a conceptual model of Sc into a computer simulation model (SM_c). The performance measure for SSc is accuracy (how close to the real system during logical modeling: f_1) and ease-of-use (the effort to build a model: f_2). The relationship of the real system, simulation model, and simulation software system is shown in the

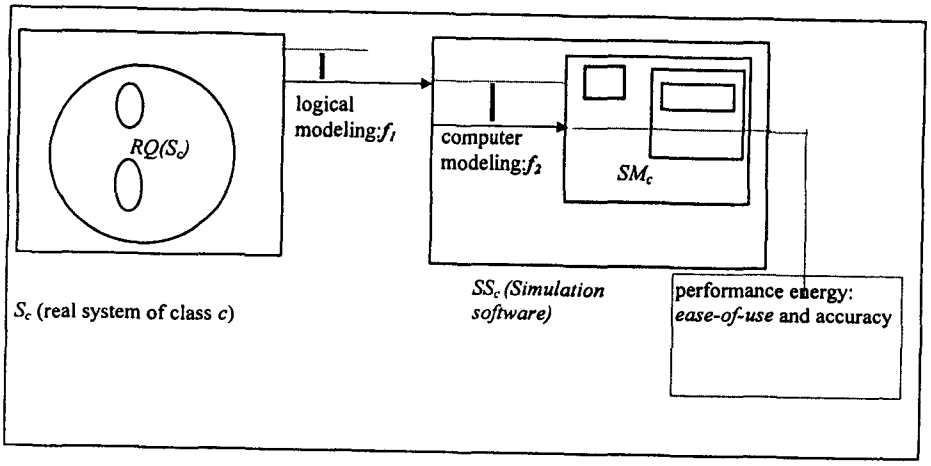

Figure 2. Relationship of real system, model, and simulation software

3.1 Within a domain user group

As a user uses a particular simulation software system, all required building blocks (or modules in technical terms) are assumed to be present for a certain class of models. This is considered to be steady state performance of SS_c. Therefore, ease-of-use and

accuracy are the performance measures for evaluating the steady state use of SS_c. The effort required to build the different instances of SM_c involves only addition and/or deletion of modules followed by modification of the data values in the modules. The following notations are used to illustrate the flexibility concept and analysis in simulation software systems under steady state use.

Notations:
M_i: the i^{th} module in a model, i =1 to TM
$REL_{i,j}$: Relationship from module i to module j.
$ATT_{i,k}$: the k^{th} attribute in the i^{th} module, k =1 to P
$BHR_{i,l}$: the l^{th} behavior in the i^{th} module, l =1 to Q
$IRF_{i,m}$: the m^{th} interface point in the i^{th} module, m =1 to S
$DAT_{i,dt}$: the dt^{th} data in i^{th} module, d_t =1 to V

Let f(X) denote the effort required for the user to perform a task X in the process of transforming and modeling, and consider that there may be an initial model to be used for creating a new model.

* User's effort to select module a and add it to a model: $f(\sigma_a^+)$
* User's effort to select module r and remove it from an existing model: $f(\sigma_r^-)$
* User's effort to change data for the i^{th} module in the model: , where V is the number of data entries to be changed in the i^{th} module. Thus, the total effort required for the user to build a new model of the system within a domain user group can be expressed as $\sum_{a=1}^{A} f(\sigma_a^+) + \sum_{i=1}^{N} \sum_{dt=1}^{V}, f(DAT_{i,dt}) + \sum_{r=0}^{R} f(\sigma_r^-)$ where A is the total number of modules to be added, R is the total number of modules to be removed from the model, N is the total number of modules whose data need to be changed. From the user's point of view, the effort to build a model is a function of the number of modules in the simulation model SM_c and complexity of each module M_i.

3.2 Different domain user groups

As a user attempts to build different classes of systems, due to performance mismatches, all required building blocks may not be available. Therefore, a SS_c developer may need to reconfigure the simulation software system from SS_c to SS_{c+1}. Therefore, the consideration should be based on the developer's perspective and the effort to reconfigure SS_c is the performance measure for evaluating changing requirements of the system. This reconfiguration effort involves adding/modifying relationships (REL), attributes (ATT), behaviors (BHR), and interfaces (IRF) for the building blocks. Once the SS_c is reconfigured, the environment is considered to be stable until another reconfiguration occurs. This process is repeated as often as necessary.

When a change occurs, SS_c is reconfigured to form SS_{c+1}, the effort to reconfigure SS_c depends on the level of flexibility available in SS_c. The effort to reconfigure SS_c requires the following steps: conceptualizing the modules that are required to build a new class of systems, creating modules (if a similar module does not exist) or modifying behaviors (if a similar behavior exists), and configuring the modules.

Step 1: Create modules.

Modeler will face two different processes in order to create modular modules.

Case 1: If similar modules are present in the simulation software system, then the effort involves modification of BHR, IRF, and ATT. Equation (3.1), (3.2), and (3.3) shows the effort required to modify BHR, IRF, and ATT respectively.

$$\sum_{i=1}^{EM} \sum_{l=1}^{Q} f(BHR_{i,l}) \quad \text{.....(3.1)}$$

Where, Q is the total number of behaviors to be built into the ith module, and EM is the number of modules involved.

$$\sum_{i=1}^{EM} \sum_{m=1}^{S} f(IRF_{i,m}) \quad \text{.....(3.2)}$$

Where, S is the total number of interfaces to be changed in the i^{th} module, and EM is the number of modules involved.

$$\sum_{i=1}^{EM} \sum_{k=1}^{P} f(ATT_{i,k}) \quad \text{.....(3.3)}$$

Where, P is the total number of attributes to be changed in the i^{th} module, and EM is the number of modules involved.

Case 2: If modules are not available in the existing software system, then new modules must be built. Effort to build new modules can be expressed as equation (3.4).

$$\sum_{i=1}^{NM} f(M_i) \quad \text{.....(3.4)}$$

Where, NM is the number of new modules to be built. (Note: A new module consists of proper BHR, IRF, and ATT)

In both cases (case 1 and 2), the developer will go through the cycle of compile and link process. Thus, the total effort to create new modules can be expressed as equation (3.5).

$$f\Big[\sum_{i=1}^{NM} f(M_i) + \sum_{i=1}^{EM} \sum_{l=1}^{Q} f(BHR_{i,l}) + \sum_{i=1}^{EM} \sum_{m=1}^{S} f(IRF_{i,m}) + \sum_{i=1}^{EM} \sum_{k=1}^{P} f(ATT_{i,k})\Big] \quad \text{.....(3.5)}$$

Step 2: Configuration of modules (i.e., establishing relationships between modules)

Effort to make relationships can be expressed as equation (3.6).

$$\sum_{i=1}^{TM} \sum_{j=1}^{TM} f(REL_{i,j}) \quad \text{.....(3.6)}$$

Where, TM is the total number of modules involved in a model. Therefore, the total effort to reconfigure the SS_c for a changing environment is a summation of all above equations. Thus, the total effort to reconfigure SS_c can be expressed as equation (3.7).

$$f\Big[\sum_{i=1}^{NM} f(M_i) + \sum_{i=1}^{EM} \sum_{l=1}^{Q} f(BHR_{i,l}) + \sum_{i=1}^{EM} \sum_{m=1}^{S} f(IRF_{i,m}) + \sum_{i=1}^{EM} \sum_{k=1}^{P} f(ATT_{i,k})\Big] + \sum_{i=1}^{TM} \sum_{j=1}^{TM} f(REL_{i,j}) \quad \text{..... (3.7)}$$

The above equation (3.7) is a measure of simulation software flexibility. To design a flexible simulation software architecture, the total effort to reconfigure SS_c should be made as small as possible. In other words, the objective of the design of flexible object oriented simulation architecture is to minimize the equation (3.7).

4. Concluding remarks

The classification of simulation languages to date is summarized. Historically, there has been a trade-off between ease-of-use and power. Simulation software flexibility, a new concept, is introduced, which addresses the problem of balance between power and ease of use. A systems modular approach is used for the analysis and design of both the simulation model and object oriented simulation software. The importance of the new concept, flexibility, in object oriented simulation software is described., followed by the development of formalism, and metrics for object oriented simulation software. According to the metrics developed, it is shown that the modeler can build any arbitrary model with less effort by using flexible simulation software.

References

[1] AutoMod, Auto Simulations Inc., Bountiful, Utah, 1994.

[2] Belanger, R. and A. Mullarney, ModSim II Reference Manual, CACI Products Company, La Jolla, CA., 1990.

[3] Cellier, Francois E., "Combined Continuous/Discrete Simulation Applications, Techniques, and Tools," Proc. of the 1986 Winter Simulation Conference, pp.24-33.

[4] Choi, Byung K. et al. "Object-Oriented Graphical Modeling of FMSs," The Int. J. of FMS, Vol., 8 , 1996, pp.159-182.

[5] Crookes, J.G. , "Simulation in 1991 - Ten years on," European Journal of Operations Research Vol. 57, 1992, pp. 305-308.

[6] Diamond, Bob, "EXTEND: A Library-Based, Hierarchical, Multi- Domain Modeling System," Proc. of the 1993 Winter Simulation Conference, pp.240-248.

[7] Fishwick, Paul A., "SimPack: Getting Started With Simulation Programming In C and C++," Proc. of the 1992 Winter Simulation Conference, pp.154-162.

[8] Frazelle, E. H., "Flexibility: A Strategic Response in Changing Times," Industrial Engineering, Mar., 1986, pp.17-20.

[9] Helsgaun, K., "DISCO- A Simulation Based Language For Combined Continuous and Discrete Simulation," Simulation, Jul., 1980, pp.1-11.

[10] Huang, Barney K. Computer Simulation Analysis of Biological and Agricultural Systems, CRC Press Inc., 1994, pp.147-155.

[11] Jackman, John and D. J. Medeiros, "A Graphical Methodology for Simulating Communication Networks," IEEE Transactions on Communications, Vol. 36, No. 4, 1988, pp.459-464.

[12] Karayanakis, Nicholas M. Computer-Assisted Simulation of Dynamic Systems with Block Diagram Languages, CRC Press Inc., 1993, pp. 2-7.

[13] Kettenis, Dirk L., "COSMOS: A Simulation Language for Continuous, discrete and Combined Models," Simulation, Jan., 1992, pp 32-41.

[14] Law, Averill M. and W. Kelton, Simulation Modeling and Analysis, McGraw Hill Book Co., New York, 1982.

[15] Lim, Ik Sung, Design of A Flexible Unified Object-Oriented Simulation System Architecture, Ph.D. Dissertation, 1996, Wayne State University, Detroit, Michigan

[16] Lomow, Greg and Dirk Baezner, "A Tutorial Introduction to Object-Oriented

Simulation and Sim++," Proc. of the 1991 Summer Computer Simulation Conference, pp.1165-1170.

[17] Luker, Paul A., Modeller: Computer-assisted modelling of continuous systems, May, 1984, Simulation, pp.205-214.

[18] Mejabi, O. O., "Object-Oriented Simulation Using Model Builder," Proc. of the 1993 Winter Simulation Conference, pp.303-307.

[19] Nance, R. E., "A History of Discrete Event Simulation Programming Languages," ACM SIGPLAN Notices, Vol. 28, No.3, 1993, pp.149-175.

[20] Najmi, Adeel and S. Stein, "Comparison of Conventional and Object-Oriented Approaches For Simulation of Manufacturing Systems," IIE Integrated Systems Conference and Society for Integrated Manufacturing Conference Proceedings, 1989, pp. 471 - 476.

[21] Pegden, C. and Deborah A. Davis, "ARENATM: A SIMAN/CINEMA- Based Hierarchical Modeling Systems," Proc. of the 1992 Winter Simulation Conference, pp.390-399.

[22] Silverman, D. and M. Stelzner, "SIMKIT++ Knowledge-Based Simulation Tools," In: Knowledge Systems's Paradigms, North-Holland Co., 1989, pp.189-196.

[23] Simple++ User's manual, Optimization of Systems and Business Process, AESOP, Aug., 1995

[24] Taylor II User's manual, 1995, F& H Simulations Inc.

[25] Ulgen, Onur M. and Timothy Thomasma, "SmartSim: An Object Oriented Simulation Program Generator for Manufacturing Systems," Int. J. of Production Research, Vol.28, No.9, 1990, pp.1713-1730.

[26] Wu, B. Manufacturing Systems Design and Analysis, Chapman & Hall, 1992, pp.216-226.

[27] Zeigler, B. P., Object-Oriented Simulation with Hierarchical Modular Models Intelligent Agents and Endomorphic Systems, Academic Press Inc., 1990.